EXHIBIT 1
Part 3 of 4

8. [2B] "comparing each input command word to a command word translation table, configured for storing for each prescribed command word a corresponding token, for identification of a matching token; and"

The token recognizer subcomponent of the parser-translator in Martinez-Guerra extracts input words from the input stream "in accordance with dictionary entries and rules encoded in grammar 17 and in accordance with the current parse state." (*Id.* at 9:35-38, *see also id.* at 11:51-55.) Martinez-Guerra explains that dictionary entries "encode information associated with tokens" (*id.* at 10:24-25), and "describe all the tokens that a parser-translator (*e.g.*, token recognizer 33 of parser-translator component 30) should recognize . . . ." (*Id.* at 14:16-18.) The '526 patent describes the "command word translation table" in similar terms: it "includes all the command words 26 that are valid according to the generic syntax." (Ex. 1001 at 3:44-45.) One of ordinary skill in the art would understand from these disclosures, and in view of Martinez-Guerra's example of the "delete /usr/extract/testing" generic command, that the dictionary entries comprise a command word translation table used by the token recognizer to match input words with valid tokens found in the phrase structure rules. (Clark Decl. ¶ 63; *see also* Ex. 1002 at 14:55-59 ("A rule specification includes a sequence of grammar categories that define a legal sequence (or sequences) of tokens in the language. Tokens (read from the input stream) 'match' categories.").)

For example, when the user submits the "delete /usr/extract/testing" command, the first input word "delete" is matched with a "regular token" entry in the dictionary having the same content ("delete"). (Clark Decl. ¶ 64.) As Martinez-Guerra explains, a "regular token is a fixed sequence of characters (for example, the string 'delete')." (Ex. 1002 at 15:2-4.) The next (and final) input word the user submits is the pathname for the folder or file to be deleted. (Clark Decl. ¶ 64; Ex. 1002 at 15:57-16:5.) Martinez-Guerra explains that "pathname-expert" is used in place of the actual pathname entered by the user (*e.g.*, "/usr/extract/testing"). (Clark Decl. ¶ 64; Ex. 1002 at 15:6-15, 15:54-16:5.) When the input word "/usr/extract/testing" is received from the user, the parser-translator uses the dictionary entries to match the original input word with "pathname-expert." (Clark Decl. ¶ 64; Ex. 1002 at 15:4-15, 15:63-16:5.) The parser-translator then uses the matching token "pathname-expert" from the dictionary entries *instead of* the original input word "/usr/extract/testing" when searching for a match in the command parse tree. (Clark Decl. ¶ 64; Ex. 1002 at 15:54-16:5.)

9. [2C] "determining a presence of the matching token within the command parse tree for each input command word."

For each input command word the token recognizer subcomponent matches to a corresponding token in the dictionary entries (Ex. 1002 at 11:51-55), it determines whether that token is among the next legal choices in the parse states maintained by the parser. (*Id.* at 18:36-42, 18:66-19:7.) As discussed above, the

parser-translator uses "internal data structures," including "parse state data structures" and "internal representations of a grammar" to specify the next legal choices allowed by the grammar, and the phrase structure rules encoded in the grammar collectively constitute a command parse tree. (Clark Decl. ¶ 65; Ex. 1002 at 14:10-15, 18:31-48.) If the parser-translator determines that a token matches an expected next legal choice in the parse state, then it adds the token to the parse state, and the parser computes an updated list of next legal choices (*i.e.*, tokens that may legally follow the matched token in the command parse tree). (Clark Decl. ¶ 65; Ex. 1002 at 19:5-35.) Stated in terms of the '526 patent, the parser-translator determines the presence of a matching token (*e.g.*, "delete") within the command parse tree. (Clark Decl. ¶ 65.)

In the example discussed in the previous limitation, the parser-translator first determines the presence of the token corresponding to the input word "delete" in the command parse tree. (Clark Decl. ¶ 66; Ex. 1002 at 15:50-62, 18:31-48.) A matching token is found within the command parse tree because "delete" is the first token in the phrase structure rule RULE1. (Clark Decl. ¶ 66; Ex. 1002 at 15:57; 20:40-45, Fig. 10.) Having matched the first token, the parser-translator then determines the presence of the second (and final) token, "pathname-expert," which corresponds to a valid file or folder pathname supplied by the user ("/usr/extract/testing"), as explained above. (Clark Decl. ¶ 66; Ex. 1002 at 15:50-

16:5.) The "pathname-expert" token is also found in the command parse tree because it corresponds to the final token of the phrase structure rule RULE1. (Clark Decl. ¶ 66; Ex. 1002 at 15:57.) The parser-translator of Martinez-Guerra thus determines the presence of a matching token in the command parse tree for each input command word.

10. [3] "The method of claim 2, wherein the determining step includes recursively traversing the command parse tree based on an order of the input command words for identification of the matching token within the identified one element."

As discussed above, the Martinez-Guerra token recognizer determines, for each input command word, the presence of a token in the phrase structure rules of the grammar, which for any finite high-level command language together constitute a command parse tree as described in the '526 patent. (Clark Decl. ¶ 67; Ex. 1002 at 18:31-40.) Martinez-Guerra teaches that "[t]oken recognizer 31 performs input matching against each of the valid next states corresponding to a parse state representation consistent with the input stream read so far." (Ex. 1002 at 20:31-34.) As successive tokens are matched to potential next legal choices (*i.e.*, "elements") in the representation of the current parse state (*id.* at 20:35-40), the parser-translator updates the current parse state. (Clark Decl. ¶ 67; Ex. 1002 at 20:42-45, 20:49-54.) This process of matching input tokens to next legal choices in the phrase structure rules is repeated for each word in the input stream, in order,

until a complete phrase, as defined by a phrase structure rule, has been parsed. (Clark Decl. ¶ 67; 20:42-48.) Thus, Martinez-Guerra discloses processing the input stream based on an order (*i.e.*, left-to-right) of the input command words. (Clark Decl. ¶ 67.)

One of ordinary skill in the art would understand that this process "recursively traverses" the command parse tree in the same way as the '526 patent. (Clark Decl. ¶ 68.) Figure 10 of Martinez-Guerra "illustrates the application of an input token to each parse state" (Ex. 1002 at 20:40-41), and this process repeats for each input token in the order received. (Clark Decl. ¶ 68; Ex. 1002 at 10:53-55, 18:62-65 ("Additional tokens are parsed from the input stream (*e.g.*, input stream 36) until a complete statement is defined in accordance with the particular grammar employed.").) The same process is used in Figure 3 of the '526 patent, where tokens are matched to elements in the command parse tree. (Clark Decl. ¶ 68.) Because Martinez-Guerra uses the same process for matching tokens to next legal choices in the command parse tree as the '526 patent, it "recursively traverses" the command parse tree as claimed.

Moreover, even if the Board were to construe this claim limitation in accordance with the mathematical meaning of "recursive"—*i.e.*, "a function that calls itself"—Martinez-Guerra still renders this limitation obvious. Skilled artisans understood at the time of the alleged invention of the '526 patent that repeating

processes and recursive processes were known alternatives for performing tasks such as those claimed, and that a recursive process could be substituted for a repeating one with predictable results. (Clark Decl. ¶ 69.) In particular, recursive processing (in the stricter sense) has long been seen as particularly well-suited to trees. (*Id.*) Accordingly, it would have been obvious to a skilled artisan to modify the teachings of Martinez-Guerra to use a recursive process (*i.e.*, one that calls itself) rather than a repeating one as both Martinez-Guerra and the '526 patent disclose.

11. [4] "The method of claim 3, wherein the issuing step includes issuing the prescribed command based on a corresponding command key specified for the matching token within the identified one element."

Martinez-Guerra discloses issuing the prescribed command based on a "translation function identifier" specified for the element that completes a phrase structure rule, such as the "pathname-expert" element of "RULE1." (Clark Decl. ¶ 70; Ex. 1002 at 15:50-62, 16:11-16.) When the input stream contains a legal ordering of words according to a phrase structure rule, the translation function corresponding to that phrase structure rule is used to determine the translation. (Ex. 1002 at 16:15-16, 17:47-52, 20:61-64.) The output of this process is the prescribed command (*e.g.*, "rm /usr/extract/testing"). (Clark Decl. ¶ 70; Ex. 1002 at 15:50-62.)

The "translation function identifier" specifies where a translation function corresponding to a matched phrase structure rule can be located. (Clark Decl. ¶ 70; Ex. 1002 at 16:11-16, 20:61-64.) It is thus a "command key," as claimed in the '526 patent. (Clark Decl. ¶ 70.) Martinez-Guerra explains that the translation function identifier "may take the form of . . . an index specifying a function number," and "allows parser-translator component 30 (particularly token recognizer 31 and translator 33 subcomponents thereof) to invoke the corresponding . . . translation function in accordance with the grammar 37." (Ex. 1002 at 16:16-24.)

Thus, in the example discussed in claim 1 above, when the final token "pathname-expert" (which is used in place of the input word "/usr/extract/testing" in the generic command "delete /usr/extract/testing" received from the user) is matched with the "pathname-expert" element in the portion of the command parse tree containing RULE1, a translation function identifier specified for RULE1 tells the parser-translator where to locate the translation function needed for the translator to issue the prescribed command "rm /usr/extract/testing." (Clark Decl. ¶ 71.) Accordingly, the prescribed command is based on a corresponding command key specified for the matching token "pathname-expert" within the identified one element.

12. [5] "The method of claim 4, wherein the issuing step further includes accessing a prescribed translator configured for converting the generic command according to the corresponding command format into the prescribed command based on the corresponding command key."

Martinez-Guerra discloses accessing a prescribed translator (*e.g.*, translator 33) that is configured for converting the generic command according to the corresponding command format (*e.g.*, "delete /usr/extract/testing") into the prescribed command (*e.g.*, "rm /usr/extract/testing") based on the corresponding command key (*e.g.*, the translation function identifier corresponding to phrase structure rule RULE1). (Ex. 1002 at 17:51-58 ("A translation function is associated with a phrase structure rule and is invoked by translator 33."); 15:50-62 (translating the generic command "delete /usr/extract/testing" to the prescribed command "rm /usr/extract/testing" based on the translation function corresponding to phrase structure rule RULE1).) Martinez-Guerra explains that, when a phrase structure rule is satisfied by the input words received, the translation corresponding to that rule is performed by a translator component of the parser-translator. (Clark Decl. ¶ 72; Ex. 1002 at 13:29-33, 15:24-28, 17:51-58, 20:61-64.) In carrying out such a translation, the prescribed translator that converts the generic command to the prescribed command is necessarily "accessed." (Clark Decl. ¶ 72.) As discussed above with reference to claim 4, the "translation function identifier" specifies where a translation function corresponding to a matched phrase structure

rule can be located ("the prescribed translator," in the words of the'526 patent). (Clark Decl. ¶ 72; Ex. 1002 at 16:11-16, 20:61-64.) The accessed translator thus performs the translation based on the translation function identifier (*i.e.*, "command key"). (*See* discussion of claim 4 above; Clark Decl ¶ 72; Ex. 1002 at 20:61-64.)

13. [6/8] "The method of claim [5/1], wherein the validating step including validating at least a portion of the generic command by identifying the one element having the best match relative to the portion of the generic command, the issuing step including issuing the prescribed command based on the identified one element corresponding to the portion of the generic command."

Claims 6 and 8 recite the same limitation, but depend respectively from claims 5 and 1. Because the claim limitations are identical, the analysis below applies to each of them.

As discussed above with reference to claim 1, the parser-translator validates the generic command "delete /usr/extract/testing" because the token recognizer determines, for each of the input words "delete" and "/usr/extract/testing" (where "/usr/extract/testing" is matched with the token "pathname-expert" in a dictionary entry), that the token corresponding to each input word is a next legal choice according to the phrase structure rule RULE1. (Clark Decl. ¶ 73.) By matching the final token ("pathname-expert") to the corresponding element in RULE1, Martinez-Guerra discloses validating at least a portion of the generic command by

identifying the one element having the best match relative to the portion of the generic command. (*Id.* ¶ 73.) Martinez-Guerra satisfies the limitation because validating the entire generic command "delete /usr/extract/testing" necessarily validates "at least a portion of" the generic command. Further, Martinez-Guerra discloses that if an input word is invalid or a command is incomplete—for example, if only the word "delete" is received without any pathname to define the target of the operation—that action may result in returning an error. (*See* Ex. 1002 at 19:3-4.) In this circumstance, the parser-translator identifies a "best match" for less than the entire command. (Clark Decl. ¶ 73.)

Martinez-Guerra further issues the prescribed command "rm /usr/extract/testing" based on the identified one element of the command parse tree ("pathname-expert") corresponding to the portion of the generic command ("/usr/extract/testing"). As discussed above with reference to claim 1, "pathname-expert" is the final element in phrase structure rule RULE1. (*Id.* ¶ 74.) Martinez-Guerra explains that once the final token in the sequence of legal tokens for a phrase structure rule is matched, the translation rule corresponding to the phrase structure rule is used by the translator to output a translation. (Ex. 1002 at 9:40-44, 15:24-28, 17:47-52, 20:61-64.) Accordingly, when the "pathname-expert" token (which is used in place of the input word "/usr/extract/testing" in the generic command) is matched with the final element "pathname-expert" of RULE1,

RULE1 is fully parsed, and the translation function corresponding to RULE1 is used to issue the prescribed command "rm /usr/extract/testing." (Clark Decl. ¶ 74; Ex. 1002 at 20:61-64, 21:1-4.) Martinez-Guerra explains that this is accomplished by the translator substituting the original value of "pathname-expert" (the input word "/usr/extract/testing") in the translation function ("rm <pathname-expert>"). (Ex. 1002 at 15:65-16:5.) The issued prescribed command is thus "rm /usr/extract/testing." (*Id.* at 16:5.)

14. [7/9] "The method of claim [6/8], further comprising executing the prescribed command within the corresponding selected one management program."

Claims 7 and 9 recite the same limitation, but depend respectively from claims 6 and 8. Because claims 7 and 9 recite the same limitation, and each depends from a claim that is rendered obvious by Martinez-Guerra as shown above, the analysis below applies to each of them.

It was obvious to one of skill in the art at the time of the alleged invention that, after the prescribed command is issued to the selected management program, the management program executes the command. As discussed with reference to claim 1, Martinez-Guerra discloses, *e.g.*, translating the generic command "delete /usr/extract/testing" into the prescribed command "rm /usr/extract/testing," which is an executable command for removing the file located at "/usr/extract/testing" using a UNIX shell program. (Clark Decl. ¶ 75; Ex. 1002 at 15:50-62.) Martinez-

Guerra's "delete /usr/extract/testing" example is intended to illustrate the translation capability of the parser-translator and, accordingly, does not expressly state that the UNIX shell program executes this command. However, persons of ordinary skill in the art at the time of the alleged invention of the '526 patent would have found it obvious from this example, Martinez-Guerra's teachings, and their own knowledge, skill, and experience in the art, that the UNIX shell program would thereafter execute the prescribed "rm" command. (Clark Decl. ¶ 75.)

Martinez-Guerra states that an objective of the invention is to allow users to "specify complex . . . transformation statements in a high-level user language, . . . and to translate [those statements]. . . into logically and syntactically correct directives *for performing the desired data transformations or operations*." (Ex. 1002 at 3:30-37; *see also* 4:4-9, 4:47-51) (emphasis added). One of ordinary skill in the art would understand from this disclosure that Martinez-Guerra intended that the translated directives ("prescribed commands") were to be executed by the software tools that receive those directives. Indeed, a translated directive must be executed if it is to "perform the desired data transformation or operation." (Clark Decl. ¶ 76.) If the command were not executed, there would be no purpose in carrying out the translation to begin with. (*Id.*)

In the example of the "delete /usr/extract/testing" command, the parser-translator translates the generic command "delete /usr/extract/testing" into the

"logically and syntactically correct directive" "rm /usr/extract/testing" for the UNIX shell program. (*Id.* at ¶ 75; Ex. 1002 at 15:50-62.) One of ordinary skill in the art would understand from Martinez-Guerra's disclosure that the purpose of such a translation is for the UNIX shell program to execute—or "perform[] the desired data transformation[] or operation[]" (Ex. 1002 at 3:36-37) according to—the prescribed command. (Clark Decl. ¶ 75.) Accordingly, these claims are invalid for obviousness.

B. Claim 10 and dependent claims 11-13 are invalid as obvious over Martinez-Guerra

1. [10A] "A system configured for executing a plurality of management programs according to respective command formats, the system comprising:"

As discussed above with reference to claim element 1A, Martinez-Guerra discloses the claimed system, wherein a parser-translator uses a "high-level user language" to interface with a plurality of software applications. (Ex. 1002, Abstract and 3:48-53.) It does so by translating statements (which can include generic commands) made in a high-level language to "logically and syntactically correct directives for performing the desired data transformations or operations" using one or more software tools (*e.g.*, prescribed commands). (*Id.* at 3:29-37.) The method is carried out by software executed on a computer system. (*Id.* at 9:10-13; *see also* Clark Decl. ¶ 79.)

As also discussed above with reference to claim element 1A, Martinez-Guerra discloses a broad range of "management programs" with which the described parser-translator may be used. (*See* Clark Decl. ¶ 80.) Petitioners incorporate by reference and respectfully refer the Board to the discussion of claim element 1A for this limitation.

2. [10B.1] "a parser having a command parse tree configured for validating a generic command received from a user,"

The parser-translator of Martinez-Guerra, which includes a parser subcomponent (*id.* at 8:18-23, 9:24-26, Figs. 1-3), is configured to validate generic commands received from the user using phrase structure rules encoded in the grammar. (Clark Decl. ¶ 81.) As discussed above with reference to claim element 1C.1, for any finite high-level command language, Martinez-Guerra's phrase structure rules—which Martinez-Guerra teaches may be represented as an "internal data structure[]," including an "internal representation[] of a grammar"—constitute a command parse tree because they are collectively a hierarchical data representation of the valid components of statements in a high-level language, whose elements specify at least one corresponding generic command component and a corresponding action value. (Clark Decl. ¶¶ 53, 81-82; Ex. 1002 at 18:31-35.)

Martinez-Guerra discloses a user interface that receives a statement from a user in a high-level language via an input stream. (Ex. 1002 at Fig. 1; 4:4-7; 9:15-

23.) The parser-translator translates statements in that language into "statements or directives appropriate to a particular data processing application." (*Id.* at 4:4-9.) One of ordinary skill in the art would understand that the high-level language described in Martinez-Guerra includes "generic commands" within the meaning of the '526 patent, *i.e.*, a command that is an abstraction of a desired operation that is later translated into a directive for a specific software tool. (Clark Decl. ¶ 81; *see also* Ex. 1002, Abstract, 10:7-11, 13:30-33.) As discussed above, for example, Martinez-Guerra discloses that the parser-translator may receive a statement with the generic command, "delete /usr/extract/testing." (Ex. 1002 at 15:50-62 ("the string delete in an input stream such as input stream 36").)

As discussed with reference to claim element 1C.1 above, Martinez-Guerra teaches that the parser builds an "internal data structure[]," including an "internal representation[] of a grammar," that identifies the "valid next states" according to phrase structure rules encoded in the grammar. (Clark Decl. ¶¶ 53, 81-82; Ex. 1002 at 18:31-48.) For a finite command language, these phrase structure rules encoded in the grammar collectively constitute a command parse tree within the meaning of the '526 patent. (Clark Decl. ¶¶ 21-24, 53, 81-82.)

As input words are received from the user by the parser-translator, they are validated according to the phrase structure rules. As Martinez-Guerra explains, the token recognizer subcomponent of the parser-translator evaluates incoming words

received from the user against these next legal choices in the phrase structure rules encoded in the grammar. (Clark Decl. ¶ 83; Ex. 1002 at 18:66-19:15; 19:25-35.) If the input word matches a next legal choice in the phrase structure rules, then it is validated because it is a permissible choice. (Clark Decl. ¶ 83; Ex. 1002 at 18:66-19:11.)

As Martinez-Guerra illustrates, if the input word "delete" is received by the parser-translator, it will match the initial (root level) legal choice for the specification of RULE1 ("delete" in the rule specification "delete pathname-expert") because "delete" is a valid initial command term. (Clark Decl. ¶ 84; Ex. 1002 at 15:50-62; *see also* 19:25-29 ("[P]arser 32 begins with a first parse state and, consistent with phrase structure rules encoded in grammar 37, collects those tokens that are potentially valid next states corresponding to the first parse state.").) Once validated, the token is accepted by the parser, and the parse state is updated to provide the token recognizer with a new list of next legal choices (*e.g.*, tokens that may permissibly follow the token "delete" in the internal data structure representing RULE1). (Clark Decl. ¶ 84; Ex. 1002 at 19:5-7, 19:16-25, 19:31-35.) If the user provides a valid pathname as the second input word, then the parser-translator validates the corresponding token as a next legal choice allowed by the phrase structure rules encoded in the grammar (*i.e.*, "pathname-expert" in the rule

specification "delete pathname-expert"). (Clark Decl. ¶ 84; Ex. 1002 at 15:59-16:5, 20:40-49.)

The parser-translator thus uses a command parse tree configured to validate a generic command received from the user.

3. [10B.2] "the command parse tree configured for specifying valid generic commands relative to a prescribed generic command format and having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value,"

As explained above with reference to claim element 1C.1, Martinez-Guerra discloses a command parse tree configured for specifying valid generic commands (*e.g.*, "delete /usr/extract/testing") relative to a prescribed command format (*e.g.*, as defined in RULE1, "delete pathname-expert." (Ex. 1002 at 15:50-62; Clark Decl. ¶¶ 55, 83-84, 86.) Each token of a phrase structure rule, such as RULE1, is what the '526 patent calls a "generic command component." (Clark Decl. ¶ 86.) As further discussed above with reference to claim element 1C.2, each of these components of the phrase structure rules corresponds to an appropriate action that will occur if that component is a "best match" for the generic command. (Clark Decl. ¶¶ 87, 89.) Such a command action value exists for each generic command component of a given phrase structure rule because, as input words are parsed, they may be invalid, resulting in an error, or valid, leading ultimately to the satisfaction